same domain to an entirely declarative planning system (ASPEN) was investigated, and, with some translation, much of the procedural knowledge encoding is amenable to declarative knowledge encoding.

The approach was to compose translators from the core languages used for adapting MAGPEN, which consists of Europa and APGEN. Europa is a constraint-based planner/scheduler where domains are encoded using a declarative model. APGEN is also constraint-based, in that it tracks constraints on resources and states and other variables. Domains are encoded in both constraints and code snippets that execute according to a forward sweep through the plan. Europa and APGEN communicate to each other using proxy activities in APGEN that represent constraints and/or tokens in Europa. The composition of a translator from Europa to ASPEN was fairly straightforward, as ASPEN is also a declarative planning system, and the specific uses of Europa for the MER domain matched ASPEN's native encoding fairly closely.

On the other hand, translating from APGEN to ASPEN was considerably more involved. On the surface, the types of activities and resources one encodes in APGEN appear to match one-to-one to the activities, state variables, and resources in ASPEN. But, when looking into the definitions of how resources are profiled and activities are expanded, one sees code snippets that access various information available during planning for the moment in time being planned to decide at the time what the appropriate profile or expansion is. APGEN is actually a forward (in time) sweeping discrete event simulator, where the model is composed of code snippets that are artfully interleaved by the engine to produce a plan/schedule. To solve this problem, representative code is simulated as a declarative series of task expansions.

Predominantly, three types of procedural models were translated: loops, if-statements, and code blocks. Loops and if-statements were handled using controlled task expansion, and code blocks were handled using constraint networks that maintained the generation of results based on what the order of execution would be for a procedural representation.

One advantage with respect to performance for MAPGEN is the use of APGEN's GUI. This GUI is written in C++ and Motif, and performs very well for large plans.

This work was done by Gregg R. Rabideau, Russell L. Knight, Matthew Lenda, and Pierre F. Maldague of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

The software used in this innovation is available for commercial licensing. Please contact Dan Broderick at Daniel.F.Broderick@jpl.nasa.gov. Refer to NPO-48597.

Support Routines for In Situ Image Processing

NASA's Jet Propulsion Laboratory, Pasadena, California

This software consists of a set of application programs that support ground-based image processing for *in situ* missions. These programs represent a collection of utility routines that perform miscellaneous functions in the context of the ground data system. Each one fulfills some specific need as determined via operational experience. The most unique aspect to these programs is that they are integrated into the large, *in situ* image processing system via the PIG (Planetary Image Geometry) library. They work directly with space *in situ* data, understanding the appropriate image meta-data fields and updating them properly. The programs themselves are completely multimission; all mission dependencies are handled by PIG.

This suite of programs consists of:

- marscahv: Generates a linearized, epipolar aligned image given a stereo pair of images. These images are optimized for 1-D stereo correlations.
- marscheckcm: Compares the camera model in an image label with one derived via kinematics modeling on the ground.
- marschkovl: Checks the overlaps between a list of images in order to determine which might be stereo pairs. This is useful for non-traditional stereo images like long-baseline or those from an articulating arm camera.
- marscoordtrans: Translates mosaic coordinates from one form into another.
- marsdispcompare: Checks a Left→Right stereo disparity image against a Right→Left disparity image to ensure they are consistent with each other.
- marsdispwarp: Takes one image of a stereo pair and warps it through a disparity map to create a synthetic opposite-eye image. For example, a right eye image could be transformed to look like it was taken from the left eye via this program.
- marsfidfinder: Finds fiducial markers in an image by projecting their approximate location and then using correlation to locate the markers to subpixel accuracy. These fiducial markets are small targets attached to the spacecraft surface. This helps verify, or improve, the pointing of *in situ* cameras.
- marsinvrange: Inverse of marsrange — given a range file, re-computes an XYZ file that closely matches the original.
- marsproj: Projects an XYZ coordinate through the camera model, and reports the line/sample coordinates of the point in the image.
- marsprojfid: Given the output of marsfidfinder, projects the XYZ locations and compares them to the found locations, creating a report showing the fiducial errors in each image.
- marsrad: Radiometrically corrects an image.
- marsrelabel: Updates coordinate system or camera model labels in an image.
- marstiexyz: Given a stereo pair, allows the user to interactively pick a point in each image and reports the XYZ value corresponding to that pair of locations.
- marsunmosaic: Extracts a single frame from a mosaic, which will be created such that it could have been an input to the original mosaic. Useful for creating simulated input frames using different camera models than the original mosaic used.
- merinverter: Uses an inverse lookup table to convert 8-bit telemetered data to its 12-bit original form. Can be used in other missions despite the name.

This work was done by Robert G. Deen, Oleg Pariser, Mathew C. Yeates, Hyun H. Lee, and Jean Lorre of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Dan Broderick at Daniel.F.Broderick@jpl.nasa.gov. Refer to NPO-47728.